Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Leon Oven

Povezljivost sistema ERP SAP z mobilnimi napravami

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Damjan Vavpotič

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V okviru diplomske naloge raziščite možnosti za integracijo sistema SAP z mobilnimi napravami. Preučite in predstavite nekaj pomembnejših obstoječih rešitev oz. programskih ogrodij za ta namen. V praktičnem delu naloge z uporabo izbrane tehnologije za povezovanje pripravite delujoč prototip mobilne aplikacije, ki se poveže s sistemom SAP. Predstavite, kako je potekal razvoj prototipa, njegovo zgradbo in kratko opišite tudi njegovo delovanje. Izpostavite morebitne tehnične težave pri integraciji s sistemom SAP.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Leon Oven, z vpisno številko **63010103**, sem avtor diplomskega dela z naslovom:

Povezljivost sistema ERP SAP z mobilnimi napravami

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Damjana Vavpotiča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 12. septembra 2014 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Damjanu Vavpotiču za vodenje, usmerjanje in pomoč pri izdelavi diplomske naloge.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Predstavitev sistema ERP SAP **3**

2.1 Osnovna predstavitev 3

2.2 Arhitektura SAP 4

3 Koncepti razvoja mobilnih aplikacij **7**

3.1 Splošen pregled 7

3.2 Avtohtone aplikacije (Native applications) 8

3.3 Aplikacije HTML5 9

3.4 Hibridne aplikacije (Hybrid applications) 9

4 Primerjava orodij za razvoj mobilnih aplikacij za sistem SAP **11**

4.1 Komercialna orodja za razvoj 13

4.2 SyBase Unwired Platform (SUP) in SAP mobile platform (SMP) . 17

4.3 SAPUI5 in OpenUI5 18

4.4 Lasten razvoj z uporabo protokola SOAP 20

5 Mobilna aplikacija za opravila v SAP **31**

5.1 Razvojno okolje Android studio 31

5.2 Koncept in opredelitev problema 32

5.3 Proces potrjevanja računov 34

5.4 Razvoj spletnih storitev v SAP 36

5.5 Razvoj aplikacije 40

6 Sklepne ugotovitve **51**

Seznam uporabljenih kratic

kratica	angleško	slovensko
ADT	Android Development Tools	razvojna orodja za Android
CSS	Cascading Style Sheets	stilna predloga spletne strani
ERP	Enterprise Resource Planning	integriran poslovni sistem
GPS	Global positioning system	sistem globalnega določanja lege
GUI	Graphic User Interface	uporabniški vmesnik
HTML	Hypertext Markup Language	označevalni jezik za oblikovanje večpredstavnostnih dokumentov
ICM	Internet Communication Manager	vmesnik za prestrezanje zahtevkov do sistema SAP
JSON	JavaScript Object Notation	oblika za izmenjavo podatkov
MVC	Model view controller	tehnologija model, pogled, krmilnik
REST	Representational State Transfer	arhitektura za izmenjavo podatkov med spletnimi storitv.
SMP	SAP Mobile platform	platforma za mobilne naprave podjetja SAP
SOAP	Simple Object Access protocol	protokol za izmenjavo strukturiranih XML sporočil
SSL	Secure Socket Layer	kriptografski protokol, ki omogoča varno komunikacijo na medmrežju
SSO	Single Sign On	enotna prijava v različne sisteme
UDDI	Universal Discovery, Description and Integration	register za iskanje spletnih storitev
WSDL	Web Service Description Language	jezik za opis spletnih storitev
XML	Extensible Markup Language	razširljiv označevalni jezik

Povzetek

Tehnološki razvoj pametnih mobilnih naprav je v zadnjem desetletju povzročil velike spremembe v delovanju družbe in podjetij. Ker je svet postal "velika vas"in razdalja ne pomeni več velike ovire v poslovnem sodelovanju, je mobilnost zaposlenih postala izredno pomembna pri njihovih vsakodnevnih zadolžitvah in opravilih. V diplomski nalogi so predstavljene možnosti integracije informacijskega sistema SAP z mobilnimi napravami. Cilj diplomskega dela je razvoj mobilne aplikacije za opravila uporabnikov v nabiralniku sistema SAP. V diplomskem delu so najprej na kratko opisane osnovne lastnosti sistema SAP in njegova arhitektura. V nadaljevanju so predstavljene vrste mobilnih aplikacij ter njihove razlike v tehnologiji in konceptu razvoja. V tretjem delu diplomske naloge se seznanimo z nekaterimi orodji za razvoj mobilnih aplikacij za integracijo s sistemom SAP. Poleg lastnega orodja podjetja SAP je predstavljeno tudi nekaj komercialnih orodij in kasneje tudi splošen princip razvoja aplikacij preko protokola SOAP. V zadnjem delu diplomske naloge je opisan primer implementacije mobilne aplikacije za platformo Android in vseh ostalih storitev sistema SAP, ki so potrebni za uspešno integracijo.

Ključne besede: SAP, mobilne naprave, razvijalec, integracije, mobilna platforma, Android, mobilne aplikacije.

Abstract

Technological development of smart mobile devices in the last decade have caused major changes in the functioning of society and business. As the world has become a "global village" and distance is no longer significant obstacle to business partnership, mobility of employees has become extremely important in their daily duties and tasks. This thesis presents the possibility of the integration of IS SAP with mobile devices. The aim of the thesis was to develop mobile application for user tasks in the SAP inbox. The thesis first briefly introduce the basics of the SAP system and his architecture. Later on are presented types of mobile applications and what are the differences in concept development and tehnology. In the third part of the thesis some of the tools for mobile applications for integration with the SAP system are presented. In addition to main tool of company SAP there are some commercial tools presented and also a general tool for the development of applications for the Android mobile platform a SOAP protocol. In the last part is presented an example of mobile applications implementation and all other services on the side of the SAP system that are necessary for successful integration.

Keywords: SAP, mobile devices, developer, integration, mobile platform, Android, mobile aplications.

Poglavje 1

Uvod

V današnjem poslovnem svetu vse stremi k doseganju vedno višjih ciljev. Podjetja neprestano izboljšujejo kakovost svojih produktov, posodabljajo opremljenost s sodobno tehnologijo in se hitro prilagajajo tržnim razmeram za približevanje tem ciljem. Ena izmed možnosti za doseganje cilja je tudi povečanje učinkovitosti in zmanjšanje stroškov. Kot ključni dejavnik tukaj pogosto nastopi informatizacija poslovanja, ta pa se najlažje doseže z vpeljavo integriranega poslovnega sistema (*angl. Enterprise Resource Planning - ERP*). Učinkovit in uporabniku prijazen sistem ERP lahko znatno pripomore k izboljšanju povezovanja procesov znotraj podjetja in večjemu nadzoru poslovanja.

Procesa tranzicija in globalizacija podjetjem omogočata poslovanje po celem svetu. Zaradi razvoja mobilnih naprav in z njimi povezanih aplikacij lahko ta podjetja omogočijo dostop zaposlenim do procesov v podjetju tudi preko mobilnih naprav. Uporabniki tako lahko preko mobilnih naprav dostopajo do podatkov in funkcionalnosti sistema. Na ta način se omogoči hitrejši prenos informacij in bolj učinkovito poslovanje.

V diplomski nalogi bomo opisali različne tipe mobilnih aplikacij in predstavili nekaj komercialnih orodij za razvoj mobilnih aplikacij za SAP. V diplomskem delu bi radi predstavili možnosti integracije sistema ERP SAP z mobilnimi napravami in nato razvili aplikacijo, s katero bo uporabnik dostopal do opravil v sistemu. Seznanili se bomo s protokolom SOAP, s pomočjo katerega bomo komunicirali med

mobilno aplikacijo in sistemom SAP. Ugotovili in analizirali bomo težave, ki se bodo pojavile pri razvoju aplikacije in pri prilagoditvi procesov v sistemu SAP za potrebe integracije obeh sistemov.

Poglavje 2

Predstavitev sistema ERP SAP

2.1 Osnovna predstavitev

Podjetje SAP, s sedežem v Walldorfu v Nemčiji, je leta 1972 ustanovilo pet nekdanjih inženirjev podjetja IBM. Kratica SAP pomeni *Systemanalyse und Programmentwicklung* – sistemske analize in razvoj programov. Podjetje trenutno zaposluje približno 66.500 ljudi, v 188 državah sveta pa ima več kot 253.500 strank [1]. SAP je trenutno s svojim istoimenskim produktom največji ponudnik sistemov ERP na svetu.

Podjetje je leta 1973 ponudilo svoj prvi produkt SAP R/1, ta je zajemal samo aplikacije za finance in računovodstvo. V naslednjih letih je podjetje nadaljevalo razvoj novih produktov in posodobitev obstoječih s podporo za večjezičnost, saj je svoje produkte želelo ponuditi tudi v tujini. Leta 1979 so predstavili produkt SAP R/2. To je bil sistem za osrednje računalnike (*angl. Mainframe*), pri katerem so podprli tudi poslovanje z materialom in planiranje proizvodnje [2].

Leta 1992 so na trg poslali produkt SAP R/3, katerega najnovejša verzija 6.0 se uporablja še danes. V naslednjih letih so podprli namestitev produkta tudi na sistemih SUN in Windows NT. Leta 1999 so ponudili novo strategijo, ki popolnoma usklajuje podjetje s svojo ponudbo, imenovano *mySAP.com*. Ta strategija združuje rešitve e-poslovanja z obstoječimi aplikacijami ERP SAP na podlagi tehnologije Web. Leta 2010 je SAP predstavil tudi produkt SAP HANA (*angl. High-performace Analytics Appliance*), ki temelji na računalništvu v pomnilniku

in računalništvu v oblaku. Zadnja leta podjetje izredno veliko investira v razvoj produktov za podporo mobilnim napravam in računalništva v oblaku.

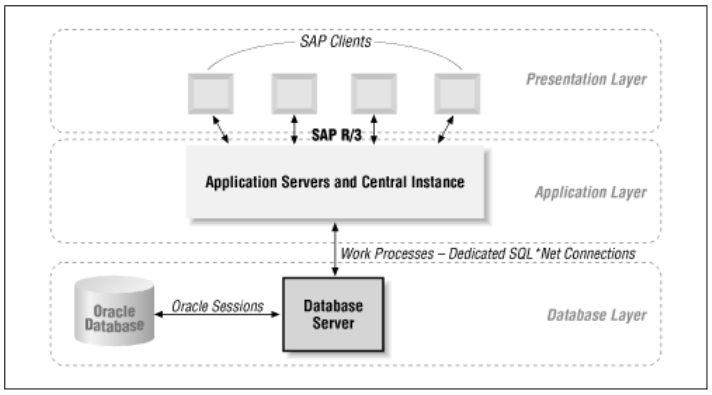
2.2 Arhitektura SAP

SAP R/3 je bil v osnovi zasnovan na tri nivojski arhitekturi (slika 2.1):

- **Uporabniški nivo** (*angl. Presentation layer*) – SAP je v tem sklopu razvil klient aplikacijo imenovano SAP GUI Client, ki je izdelana v dveh tehnologijah: za operacijski sistem Windows je razvita v jeziku C++ in nudi podporo tehnologiji .NET, za preostale operacijske sisteme pa v jeziku Java.

- **Aplikacijski nivo** (*angl. Application layer*) – na aplikacijskem nivoju vsebuje centralno instanco (*angl. Central instance*), ki je kombinacija strojne in programske opreme. Sestavljajo ga fizični strežnik (aplikacijski strežnik) in številne komponente programske opreme, vključno s strežnikom za sporočila, prehod zbirke podatkov (vnaprej določenih povezav med SAP in bazo), ter različne posodobitvene, vrstilne in dialogne programske rešitve. V večini generičnih sistemih SAP je lahko več aplikacijskih strežnikov, vendar samo ena centralna instanca.

- **Podatkovni nivo** (*angl. Database layer*) – podatkovni nivo skrbi za komunikacijo med aplikacijskim nivojem in podatkovno bazo [3].



Slika 2.1: Tri nivojska arhitektura SAP R/3, Vir slike: [3]

Zaradi potrebe po podpori čim večjemu številu podatkovnih baz so želeli razviti sistem, ki bi popolnoma ločil aplikacije od baznega nivoja. V ta namen so pripravili štiri nivojsko arhitekturo strežnik – odjemalec. SAP je izjemno kompleksen sistem, zato v tej nalogi ne bomo opisali vseh podrobnosti posameznih komponent. Vendar moramo za potrebe te naloge omeniti še *Internet Comunication Manager (ICM)*. Ta prestreza vse zahteve, ki pridejo preko interneta (tudi klici spletnih storitev) in jih nato posreduje aplikacijskemu strežniku v obdelavo.

Poglavje 3

Koncepti razvoja mobilnih aplikacij

3.1 Splošen pregled

Obstajajo različni pristopi za razvoj mobilne aplikacije, glede na te lahko razdelimo aplikacije na naslednje tipe [4]:

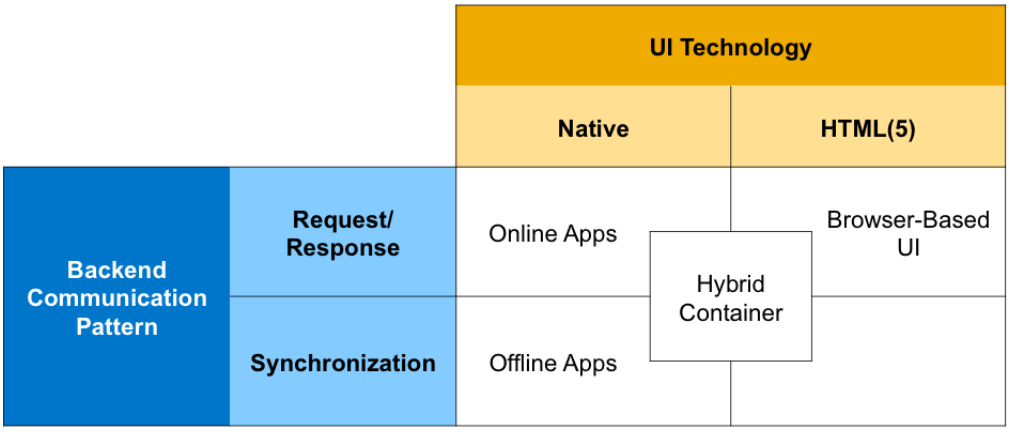
- avtohtone;
- spletne;
- hibridne.

Delitev na podlagi izbire arhitekture ni edina. Pri podjetju SAP so aplikacije razdelili na podlagi dveh vidikov [5]:

- **Tehnologija uporabniškega vmesnika** (*angl. User interface tehnology*) Tehnologijo uporabniških vmesnikov danes v splošnem delimo na dve pomembni vrsti: avtohtono tehnologijo (*angl. Native*), ki je zagotovljena z mobilno platformo, in HTML v kombinaciji z JavaScript in Cascading Style Sheets (CSS).
- **Komunikacija v ozadju** (*angl. Back-end communication*) Mobilna infrastruktura, ki je na danes na voljo, podpira dva modela za komunikacijo

med mobilnim odjemalcem in sistemom v ozadju. Prvi model je zahtevek - odgovor (*angl. request - response*), pri katerem odjemalec pošlje zahtevo sistemu v ozadju in na podlagi tega prejme odgovor. Drugi model uporablja sinhronizacijo podatkov (replikacija), pri katerem se podatki prenašajo med odjemalcem in sistemom v ozadju pred ali po tem, ko je bila aplikacija uporabljena.

Obe razdelitvi nista ločeni, ampak sta v svojih lastnostih zelo prepleteni. Sestavimo lahko shemo (slika 3.1), ki predstavlja vse možne pristope za razvoj med njima. Vsak pristop ima svoje prednosti in slabosti in ni pristopa, ki bi bil univerzalno najboljši. Sama izbira pristopa za razvoj je odvisna od posameznega primera, za katerega se aplikacija razvija.



Slika 3.1: Kombinacije mobilnih aplikacij, Vir slike: [5]

3.2 Avtohtone aplikacije (Native applications)

Avtohtona mobilna aplikacija [6] je aplikacija, ki je napisana v določenem programskem jeziku. Za android se uporablja programski jezik Java. Uporabniki so navajeni na uporabniški vmesnik, ki jim ga ponuja operacijski sistem, in funkcionalnost svoje naprave, zato pričakujejo od avtohtonih aplikacij enako uporabniško izkušnjo. Ena izmed prednosti avtohtonih aplikacij je, da lahko do potankosti izkoristi vse možnosti in zmogljivosti mobilne naprave. Aplikacija lahko dostopa do

funkcionalnosti posamezne naprave, npr. kamera, imenik, GPS itd. Slabost avtohtonih aplikacij pa je predvsem drag razvoj, saj je ta določen za eno platformo. Tipični primeri avtohtonih aplikacij so video igre. Te vrste aplikacij lahko, glede na komunikacijo s sistemom v ozadju, delimo na dva dela:

- **Aplikacije s sprotno izmenjavo podatkov** (*angl. Online applications*) Pri teh aplikacijah se izmenjava podatkov izvršuje ad-hoc in je sprožena iz mobilne aplikacije (odjemalec).
- **Aplikacije z izmenjavo podatkov v zakasnitvi** (*angl. Offline applications*) Aplikacija podatkov na mobilno infrastrukturo ne pošlje takoj, ampak jih upravlja lokalno. Šele ko zazna hitro in zanesljivo povezavo, npr. brezžična povezava, sistem vse lokalno shranjene podatke (*angl. Cached data*) pošlje na strežnik.

3.3 Aplikacije HTML5

Aplikacije HTML5 [7] so nameščene na posamezni mobilni napravi, vendar tečejo na strežniku. V bistvu so to spletne aplikacije, ki so prilagojene mobilnim napravam. Razvite so s tehnologijo HTML5 (*angl. Hyper text markup language*). Velika prednost aplikacij HTML je upravljanje na enem mestu, saj se razvijalcem ni treba ukvarjati s strategijo nameščanja. Z minimalnimi prilagoditvami je na dokaj preprost način omogočena uporaba že obstoječih spletnih aplikacij na mobilnih napravah. Za te vrste aplikacij je značilno, da imajo omejen dostop do funkcionalnosti naprave in so v veliki meri odvisne od interneta.

3.4 Hibridne aplikacije (Hybrid applications)

Hibridne aplikacije [8] so aplikacije, ki združujejo funkcionalnosti avtohtonega in spletnega razvoja. Aplikacija se namesti, zaganja in uporablja tako kot avtohtone, vendar je razvita kot spletna, se pravi v tehnologiji HTML5 z dodatki (Javascript in CSS). Aplikacija se izvaja na mobilni napravi, vendar ne v klasičnem brskalniku, temveč izkorišča avtohtono jedro za brskalnike. Take vrste aplikacije se z minimalnimi posegi lahko prilagodijo na več platform, zaradi česar so pogosto poimenovane

tudi medplatformske. Cilj razvoja medplatformske aplikacije je razviti največ skupne kode, medtem ko se ločeno za vsako platformo zagotavljata avtohton videz in edinstvena izkušnja.

Poglavje 4

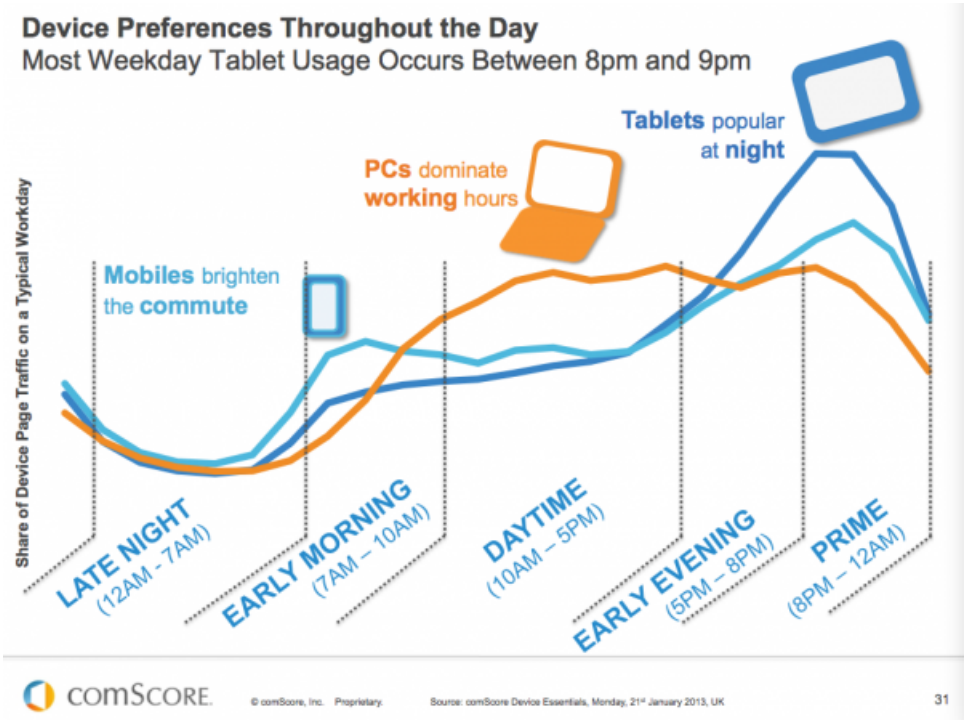
Primerjava orodij za razvoj mobilnih aplikacij za sistem SAP

Po podatkih raziskave podjetja Comscore leta 2013 [9] (slika 4.1) trenutno v delovnem času na internetu še vedno prevladujejo uporabniki osebnih računalnikov. Po njihovih napovedih naj bi se to spremenilo do leta 2015, ko naj bi prevladovali uporabniki mobilnih naprav. S popularizacijo uporabe mobilnih naprav v poslovne namene se pojavlja čedalje večja potreba tudi po mobilnih aplikacijah, s katerimi bi lahko uporabniki izvajali vsaj del dnevnih opravil iz sistema SAP.

Ker matično podjetje na začetku nastajanja mobilnih aplikacij za SAP ni ponudilo ravno najširše podpore v smislu ponudbe lastnih končnih produktov in produktov za razvoj, so se na trgu pojavili različni komercialni produkti, ki zagotavljajo podporo razvoju aplikacij za SAP. V tem poglavju bomo poskusili predstaviti nekaj glavnih ponudnikov in opisati glavne razlike med njimi. Prav tako bomo predstavili komercialni produkt podjetja SAP, znan pod imenom SAP Mobile Platform (SMP).

Vse do konca leta 2013 ni obstajala odprtokodna rešitev za razvoj mobilnih aplikacij za SAP. Kot prvi je rešitev z imenom OpenUI5 ponudilo prav podjetje SAP.

Ker pa omenjena odprtokodna rešitev deluje samo na sistemih z najnovejšimi popravki[1], smo se odločili za razvoj mobilne aplikacije za android z uporabo okolja Android Studio in protokola SOAP, ki ne predstavlja dodatnih stroškov za licence samega produkta. Seveda tak razvoj zahteva nekoliko več časa in truda, vendar za podjetja, ki nimajo potreb po vpeljavi 10 ali 20 mobilnih aplikacij, to predstavlja neznansko manjši strošek kot pa nakup ustrezne licenčne razvojne platforme.



Slika 4.1: Prikaz uporabe naprav za priklop na internet po urah v dnevu. Vir slike: [9]

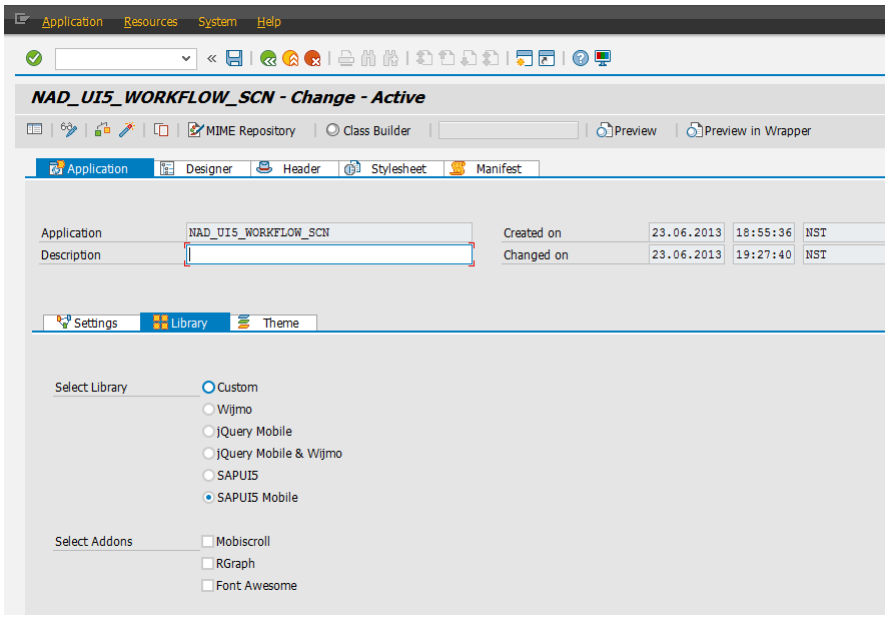
[1]Podjetja se zaradi dejstva, da je posodobitev sistema draga in da imajo najnovejši popravki izredno veliko napak, ne odločajo za takojšnje posodobitve sistemov.

4.1 Komercialna orodja za razvoj

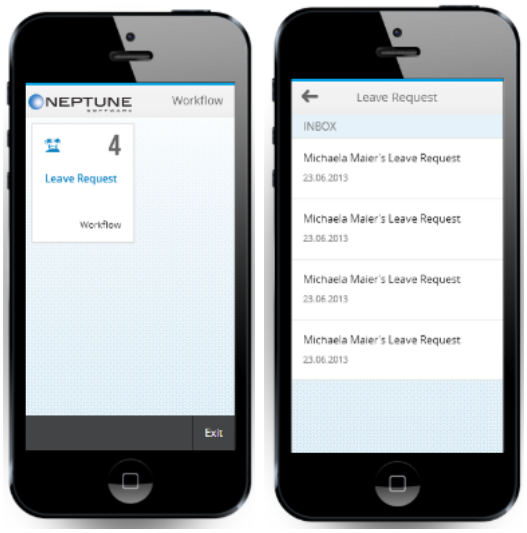
4.1.1 Neptune

Podjetje Neptune software [10] ponuja rešitev Neptun Application Designer (slika 4.2). Prednost njihove rešitve je v tem, da ni potreben noben dodaten strežnik, ki bi služil kot nekakšen vmesni člen (*angl. Middle layer*) za dostop do sistema SAP. Rešitev je razvita v razvojnem jeziku ABAP [11], s čimer so svojim potencialnim kupcem omogočili preprosto vzdrževanje in podporo.

Rešitev ponuja bogato paleto uporabniških izkušenj, kar so pri Neptune software dosegli z uporabo standarda HTML5 in ogrodij jQuery, jQuery Mobile, PhoneGap in Wijmo. Rešitev sloni na platformi Adobe PhoneGap [12]. Podjetjem na njihovem sistemu omogoča izrabo že obstoječe izvorne kode ABAP. Prav tako je treba omeniti, da za razvoj ni potrebno znanje za HTML, jQuery idr., temveč samo znanje programskega jezika ABAP, kar je velika razlika v primerjavi s konkurenčnimi produkti, pri katerih potrebujejo strokovnjake z obeh področij. Za prenos podatkov iz zalednega sistema rešitev uporablja avtohton JSON (*angl. native JSON*) [13] in zbira podatke direktno iz atributov ABAP, kar po njihovem mnenju predstavlja najboljše performance.



Slika 4.2: Pogled na razvijalsko orodje Application Designer, Vir slike: [10]



Slika 4.3: Intuitivni videz aplikacije za službeni izhod. Vir slike: [10]

4.1.2 Movillizer

Podjetje Movillizer je leta 2007 predstavilo produkt Movillizer Enterprise Mobile Platform [14]. V primerjavi s podjetjem Neptun se njegova mobilna platforma ne izvaja na sistemu SAP, ampak nudi dobro integracijo za izmenjavo podatkov. Poleg sistema ERP SAP podpira njihov produkt tudi preostale sisteme ERP, kot so Oracle, Microsoft, Salesforce idr. Arhitektura razvoja aplikacij je razvidna s slike 4.4. Njihov produkt podpira razvoj vseh vrst aplikacij: avtohtone, HTML in hibridne. Posebej velja omeniti možnost razvoja hibridnih aplikacij preko njihove platforme. Glede na vrste gradnika jih lahko razdelimo na tri tipe aplikacij:

- 100% HTML5 aplikacijo, ki se izvaja znotraj klienta Movillizer;
- aplikacija, pri kateri so nekateri ekrani HTML5, drugi pa medplatformsko avtohtoni;
- aplikacija, pri kateri je lahko na istem ekranu nekaj elementov HTML5 in nekaj elementov avtohtonega medplatformskega videza.

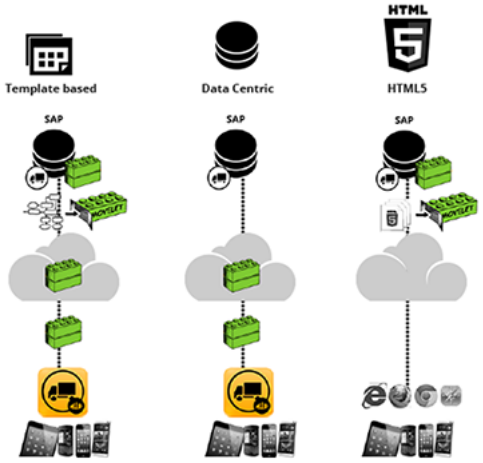
Osnovni gradnik aplikacij podjetja Movillizer je Movelet, ki vsebuje metapodatke in podatke o procesu. Splošno gledano lahko rečemo, da je Movelet neko zaporedje ekranov, ki nudijo interakcijo z uporabnikom. Aplikacija je sestavljena iz enega ali več Moveletov. Posebna lastnost Moveletov je, da jih lahko prenašamo med različnimi platformami in da se na vsaki platformi izvaja v avtohtoni kodi. Za njihovo pravilno izvajanje lahko skrbi njihov klient Movillizer (slika 4.5) ali pa so predstavljeni kot storitev v oblaku. Njihov klient podpira veliko različnih platform, to so iOS, Android, BlackBerry, Windows Mobile, Windows Phone 8, Windows 8, Windows 8 RT, J2ME, in osebni računalniki (Win32, Linux, MacOS), lahko pa se izvaja tudi kot klient v brskalniku.

Podjetje je razvilo tudi nekaj produktov za uporabo v povezavi s sistemom SAP. Ti produkti so:

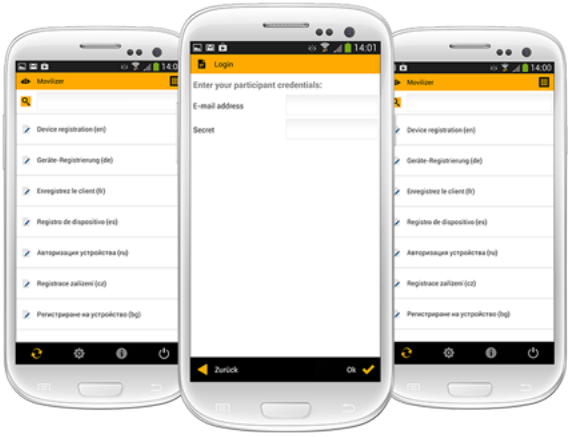
- Movillizer SAP Connector – omogoča razvoj aplikacij preko vmesnika SAP GUI;

- Movillizer za SAP PM/CS – mobilna aplikacija za podporo procesa proizvodnje;

- Movillizer za SAP MM – mobilna aplikacija za podporo materialnemu poslovanju.



Slika 4.4: Arhitektura razvoja rešitve podjetja Movillizer, Vir slike: [14]



Slika 4.5: Klient aplikacija Movillizer, Vir slike: [14]

4.2 SyBase Unwired Platform (SUP) in SAP mobile platform (SMP)

Ker je podjetje SAP znano po svoji inovativnosti, ni bilo treba dolgo čakati, da je naznanilo prihod svoje mobilne platforme. V ta namen je podjetje leta 2010 investiralo v nakup podjetja SyBase, s katerim so hitro ponudili svoj prvi produkt za razvoj aplikacij za mobilne naprave imenovan SyBase Unwired Platform (SUP). Podjetje SyBase je produkt pod enakim imenom poslalo na trg že leta 2008, vendar so nato s prihodom podjetja SAP v lastniške okvirje izvedli nekaj korenitih sprememb. Te spremembe so jim prinesle izreden uspeh, saj jih je leta 2013 Gartner določil kot vodilne na področju razvoja mobilne platforme [15] in jih tudi uvrstil v magični kvadrant za upravljanje z mobilnimi napravami (*angl. magic quadrant for mobile device management*) (slika 4.6) [16].

Slika 4.6: Gartnerjev magični kvadrant za upravljanje z mobilnimi napravami. Vir slike: [15]

Proti koncu leta 2013 so produkt popolnoma prenovili in ga ponudili pod imenom SAP Mobile Platform 3.0. Zaznamujejo ga naslednje lastnosti:

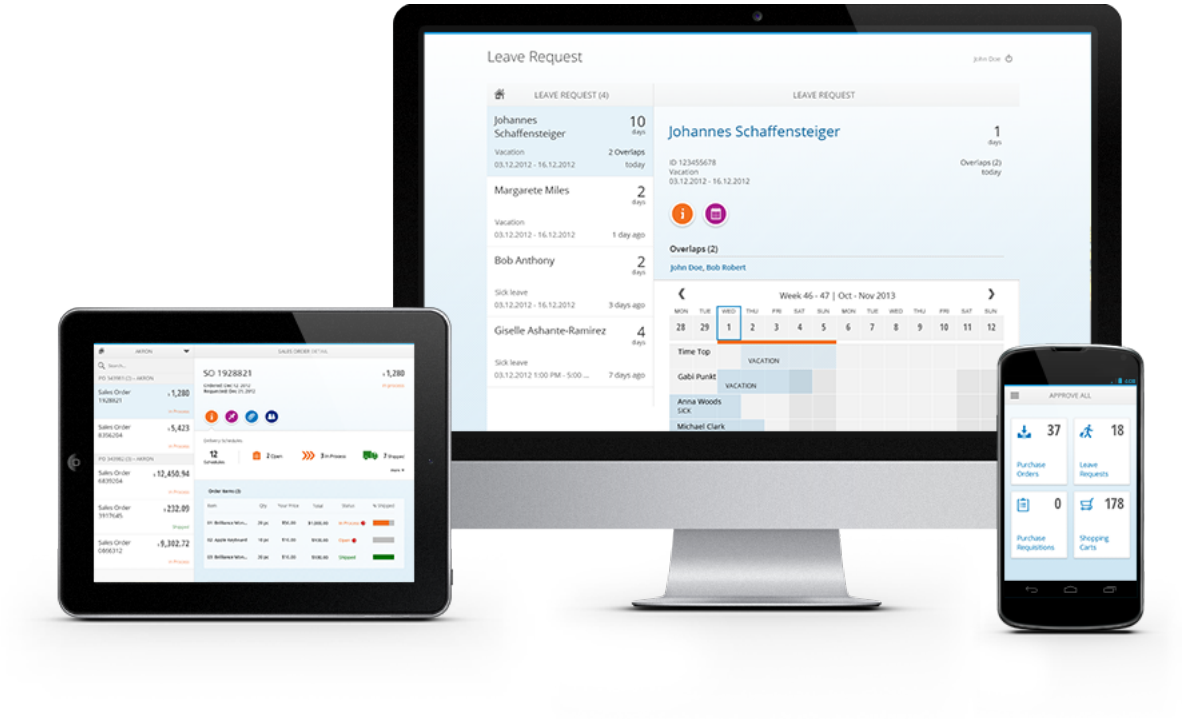
- odprtokodni standardi (*angl. Open standards*):
  - Protokol OData [17]
    Protokol OData je protokol, preko katerega se pranašajo podatki iz zalednega sistema na mobilno napravo. Protokol uporablja strukturo protokola AtomPub kot ovojnico okrog podatkov, vrnjenih iz vira. Vsi zahtevki morajo biti tipa REST;
  - arhitektura REST.
- poenotena platforma (*angl. Unified platform*)
  Poenotili so razvojna okolja SyBase Unwired Platform, Syclo Agentry in Sybase Mobillizer. Nova platforma sloni na javanskem serverju in platformi OSGI. Vse komponente nove mobilne platforme so predstavljene kot paketi OSGI, kar zagotavlja prednost pri posodabljanju, saj pred vsakim posegom ni treba ustavljati celotnega sistema;
- lahki strežnik (*angl. Lightweight server*)
  Mobilna platforma ne bo vsebovala nobenih dupliciranih ali repliciranih podatkov, saj so se v preteklosti pojavile velike težave z njihovim osveževanjem v realnosti;

4.3 SAPUI5 in OpenUI5

Podjetje SAP je že od leta 2012 razvijalo platformo za razvoj spletnih aplikacij za SAP pod imenom Phoenix, ki se je tekom razvoja preimenoval v SAPUI5. Je zbirka knjižnic HTML5 za upodabljanje in razvijanje uporabniškega vmesnika, ki bazira na tehnologiji JavaScript. Produkt uporablja odprtokodno knjižnico jQuery in podpira uporabo tehnologije CSS3 [19]. S tem produktom želijo v podjetju SAP poenotiti videz aplikacij na vseh možnih napravah (pametni telefoni, tablični računalniki in namizni računalniki). S tem produktom podjetje meri na razvijalce v njihovem sistemu[2] in stranke, ki imajo znanje spletnega razvoja. Primer razvoja

[2]Razvijalci, ki razvijajo v jeziku ABAP.

rešitve s pomočjo produkta SAPUI5 je tudi nabor rešitev po imenom FIORI UX (slika 4.7), ki so ga naznanili na konferenci Sapphire v Orlandu.



Slika 4.7: Poenoten videz aplikacij SAP preko FIORI UX. Vir slike: [20]

Produkt SAPUI5 lahko uporabljajo samo podjetja, ki so kupila določene produkte podjetja SAP. V želji predstaviti njihov produkt čim širšemu občinstvu, se je podjetje konec leta 2013 odločilo, da obstoječi produkt ponudijo tudi kot odprtokodno verzijo rešitve pod imenom OpenUI5. Razlik med produktoma v osnovnih (*angl. core*) funkcionalnostih ni veliko, naj omenimo dve največji:

- določene kontrolne knjižnice (še) niso bile označene kot odprtokodne;
- podpora odprtokodni verziji ni zagotovljena, čeprav bodo popravki, ki se nanašajo na obe verziji, dostopni tudi v odprtokodni različici.

Produkt OpenUI5[3] predstavlja resno konkurenco obstoječim uveljavljenim knjižnicam za razvoj spletnih aplikacij, npr. Bootstrap, jQuery idr. Omenimo nekaj glavnih značilnosti [19]:

[3]Od tu naprej vse opisane lastnosti OpenUI5 veljajo tudi za SAPUI5.

- **katerikoli zaslon na katerikoli napravi** (*angl. any screen on any device*) Namizje in mobilne aplikacije temeljijo na tehnologiji MVC in uporabljajo isto temeljno knjižnico;
- **brezčasna poraba podatkov SAP** (*angl. timeless SAP data consumption*) SAPUI5 loči poslovno logiko od uporabniškega vmesnika. To doseže z implementacijo REST protokola OData, ki optimizira komuniciranje s sistemom v ozadju[4];
- **odprtost in fleksibilnost** (*angl. openness and flexibility*).

OpenUI5 predstavlja resno alternativo razvojnim orodjem za izdelavo aplikacij HTML za mobilne naprave. Če upoštevamo, da je bila njegova integracija s SMP zasnovana že v začetku razvoja, lahko ugotovimo, da skupaj predstavljata resno konkurenco preostalim komercialnim produktom.

Največja ovira, zaradi katere se nismo odločili uporabiti produkta OpenUI5, je bil dostop do podatkov sistema SAP. OpenUI5 podpira dostop do podatkov preko protokola OData, ki pa na našem razvojnem strežniku zaradi licenčne politike med odločitvijo izbire tehnologije še ni bil omogočen.

4.4 Lasten razvoj z uporabo protokola SOAP

Večina orodij, s katerimi poenostavimo razvoj mobilnih aplikacij za sistem SAP, je za podjetja relativno draga. Tako smo poskušali najti alternativno rešitev, ki bi bila cenovno sprejemljiva tudi za podjetja, ki si dragih komercialnih rešitev ne more privoščiti. Kot alternativno rešitev lahko uporabimo OpenUI5, ki pa je v osnovi namenjen bolj za upodabljanje uporabniških vmesnikov za hibridne aplikacije. Alternativne odprtokodne rešitve za avtohton razvoj trenutno ne obstajajo.

Največja prepreka OpenUI5 za širšo uporabo je, da mora sistem SAP imeti najnovejše popravke, kar podjetjem predstavlja strošek in negotovost nad zanesljivostjo

[4]Podpora protokola OData v sistemu SAP je bila omogočena podjetjem brez produkta SMP šele z zadnjimi popravki.

najnovejših posodobitev. Tako se na koncu izkaže, da je za podjetja najcenejši razvoj, pri katerem se za komuniciranje s sistemom SAP uporablja protokol SOAP.

4.4.1 SOAP

SOAP je razširljiv protokol za izmenjavo strukturiranih informacij pri izvajanju spletnih storitev v računalniških omrežjih, ki za definiranje sporočil uporablja strukturo XML [18]. SOAP se uporablja za komunikacijo med aplikacijami, ki se izvajajo na različnih operacijskih sistemih ter uporabljajo različne tehnologije in programske jezike. Standardiziran je postal leta 2003 [21]. Za izmenjavo podatkov SOAP podpira skoraj vse danes poznane prenosne protokole: JMS, SMTP, TCP, UDP in HTTP. Danes se zaradi svoje podprtosti pri spletnih brskalnikih in strežnikih najpogosteje uporablja protokol HTTP (pri spletnih storitvah z večjo zahtevano varnostjo pa se uporablja HTTPS, ki takšno varnost zagotavlja s kriptiranim prenosnim protokolom).

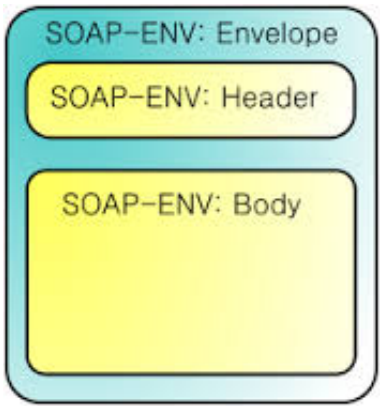
Protokol SOAP sestavljajo tri glavne karakteristike:

- **razširljivost** – pri razvoju spletne storitve lahko z razširitvami povečamo varnost, usmerjanje (*angl. routing*) in zanesljivost; namizne in mobilne aplikacije, temeljijo na MVC tehnologiji in uporabljajo isto temeljno knjižnico;
- **nevtralnost** – podpira ga večina prenosnih protokolov;
- **neodvisnost** – neodvisen od platforme.

Spletni vmesniki, ki komunicirajo preko sporočil SOAP, so definirani s pomočjo jezika WSDL (*angl. Web Service Description Language*). Da bi odjemalci lahko našli in uporabili ponudnikove spletne storitve, je potreben register oz. imenska storitev, s katero ponudniki registrirajo komponente in s tem omogočijo uporabo odjemalcem. Register se imenuje UDDI (*angl. Universal Discovery, Description and Integration*) in tudi temelji na jeziku XML, v njem pa hranimo WSDL in ostale pomembne dokumente, ki opisujejo spletno storitev in so pomembni za odjemalca [22].

Sporočilo SOAP

Sporočilo SOAP, katerega struktura je prikazana na sliki 4.8, vsebuje ovojnico (*angl. Envelope*), zaglavje (*Header*) in telo (*Body*). Telo je najpomembnejši del in vsebuje vsebino sporočila. Zaglavje je opcijsko in ponuja možnost, da v sporočilo vključimo dodatne podatke, ki dopolnjujejo vsebino telesa. Tipičen primer so kontekstne informacije, npr. varnostni kontekst. Ovojnica je korenski element sporočila ter ovija zaglavje in telo. Za sporočanje napak se uporablja element napaka (*Fault*), ki je podelement telesa.



Slika 4.8: Prikaz strukture sporočila SOAP. Vir slike: [18]

Sporočilo SOAP sestavljajo naslednji elementi:

- **ovojnica (<Envelope>)** Pomemben del ovojnice je deklaracija imenskega področja, ki označuje verzijo protokola, ki ga uporablja. Končne točke, ki prejmejo sporočilo SOAP z neveljavnim imenskim področjem, morajo sporočilo zavrniti in signalizirati napako [23];
- **telo (<Body>)** Telo vsebuje uporabne podatke. Ti podatki so lahko karkoli, od nakupa delnic, naročila za nov telefon, do kakršnegakoli elementa XML;
- **zaglavje (<Header>)** Element zaglavje se uporablja v zahtevnejših scenarijih, v katerih se želi ustvariti verigo povezanih storitev. V zaglavje se ponavadi vključuje informacije, ki vsebinsko ne sodijo v telo sporočila, npr. varnostne, transakcijske idr. [24].

```
POST /InStock HTTP/1.1
Host: www.testnadiploma.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-
    envelope">
  <soap:Header> </soap:Header>
  <soap:Body>
    <m:Zmnozi xmlns:m="http://www.testnadiploma.org/Zmnozi">
      <m:Operator1>10</m:Operator1>
      <m:Operator2>45</m:Operator2>
    </m:Zmnozi>
  </soap:Body>
</soap:Envelope>
```

Seznam 4.1: Primer sporočila SOAP

Web Services Description Language (WSDL)

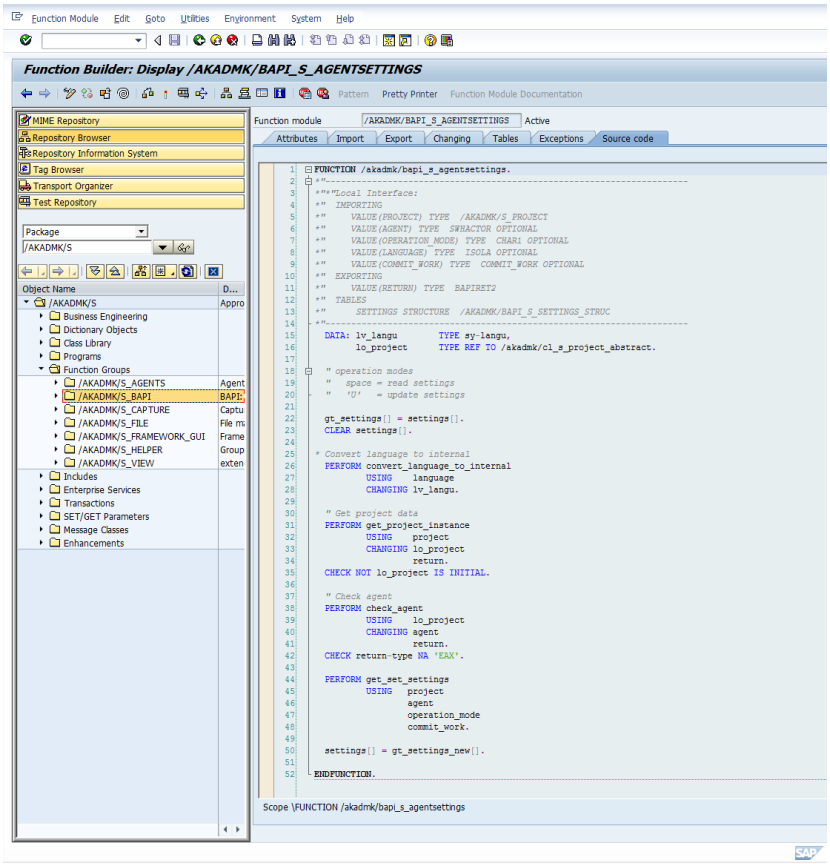
WSDL je jezik, namenjen opisovanju spletnih storitev. Tako kot protokol SOAP tudi WSDL temelji na jeziku XML. WSDL opis spletne storitve je dokument XML [24], katere primer je prikazan na seznamu 4.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap
    ="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:http="http://
    schemas.xmlsoap.org/wsdl/http/" targetNamespace="urn:sap-
    com:document:functions">
<message name="BAPI_GET_PERSONInput">
<part name="parameters" element="s0:BAPI_GET_PERSON"></part>
</message>
<message name="BAPI_GET_PERSONOutput">
<part name="parameters" element="s0:BAPI_GET_PERSON.Response"></
    part>
</message>
<portType name="BAPI_GET_PERSONPortType">
<operation name="BAPI_GET_PERSON">
<input message="s0:BAPI_GET_PERSONInput"></input>
<output message="s0:BAPI_GET_PERSONOutput"></output>
</operation>
</portType>
<binding name="BAPI_GET_PERSONBinding" type="
    s0:BAPI_GET_PERSONPortType">
<soap:binding style="document" transport="http://schemas.xmlsoap
    .org/soap/http"></soap:binding>
<operation name="BAPI_GET_PERSON">
<soap:operation soapAction="http://www.sap.com/BAPI_GET_PERSON">
    </soap:operation>
<input><soap:body use="literal"></soap:body></input>
<output><soap:body use="literal"></soap:body></output>
</operation></binding>
<service name="BAPI_GET_PERSONService">
<port name="BAPI_GET_PERSONPortType" binding="
    s0:BAPI_GET_PERSONBinding">
<soap:address location="http://test:8001/sap/bc/soap/rfc"></
    soap:address>
</port>
</service></definitions>
```

Seznam 4.2: Primer dokumenta WSDL

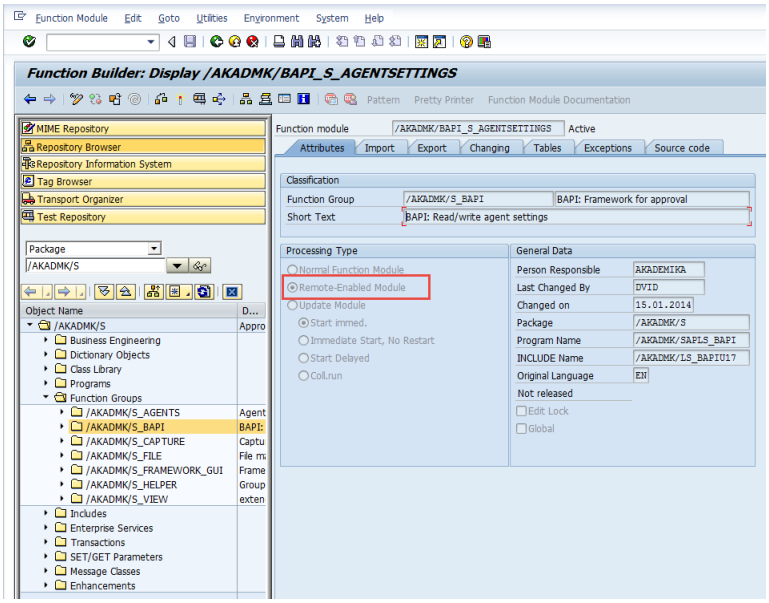
4.4.2 SOAP in SAP

Aplikacijski server SAP ima od verzije 4.6 vgrajeno podporo za spletne storitve. Vse do verzije SAP 6.0 so podpirali samo SOAP od verzije 1.0 pa do 1.2, zato je bila kreacija spletne storitve precej preprosta. V sistemu SAP se lahko kot spletni servis objavi tako imenovani funkcijski modul. Funkcijski moduli so procedure, ki so napisane v programskem jeziku ABAP. Ti moduli, katerih primer je prikazan na sliki 4.9, lahko uporabljajo preddefinirane globalne funkcije sistema SAP, vendar jih ne morejo spreminjati [25]. Funkcijski moduli igrajo pomembno vlogo pri posodabljanju in interakciji med različnimi sistemi SAP ali med SAP in zunanjim sistemom. Moduli se definirajo v posebnem orodju, imenovanem Function Builder, ki preprečuje nekompatibilne spremembe na obstoječih modulih.



Slika 4.9: Primer funkcijskega modula znotraj orodja Function Builder

Po kreaciji funkcijskega modula je ta uporaben samo znotraj okolja SAP. Če želimo modul objaviti kot spletno storitev, moramo znotraj orodja Function Builder pod zavihkom Attributes označiti Remote-Enabled Module, tako kot to prikazuje slika 4.10.



Slika 4.10: Funkcijski modul kot spletni servis

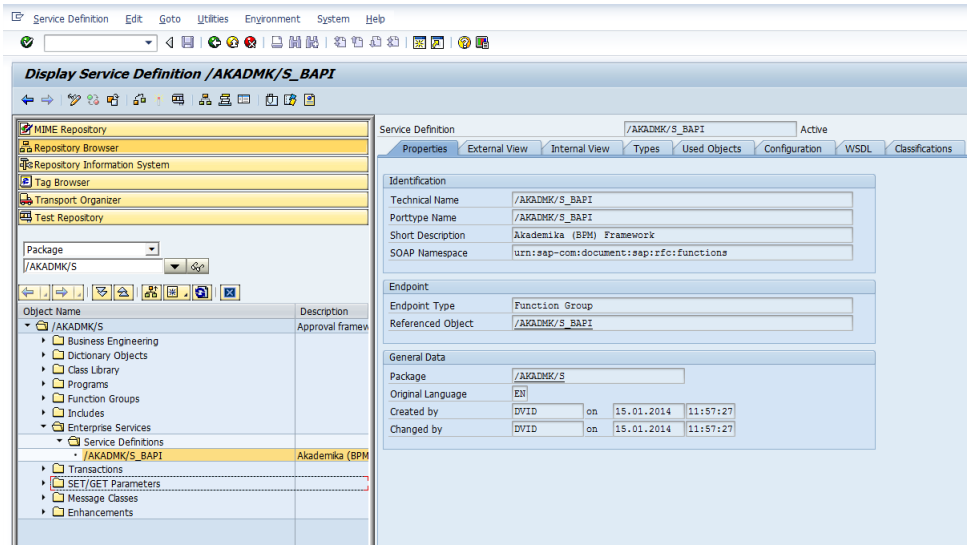
Po objavi funkcijskega modula kot spletno storitev lahko programerji dostopajo do datoteke WSDL preko aplikacije Bussines server page, imenovane WebService-Browser. Dostop do te aplikacije je mogoč preko posebnega URL naslova, katerega format prikazuje seznam 5.3 in ki ga sestavljajo naslednji strežniško odvisni elementi:

- **<ime_streznika>** – ime strežnika, na katerem teče SAP;
- **<stevilka_vrat>** – številka vrat za dostop do aplikacije;
- **<klient>** – številka klienta za dostop.

```
http://<ime_streznika>:<stevilka_vrat>/sap/bc/bsp/sap/
    WebServiceBrowser/search.html?sap-client=<klient>
```

Seznam 4.3: Format naslova za dostop do objavljenih spletnih storitev

Z verzijo SAP 6.0 pa je prišla popolnoma drugačna logika kreiranja spletnih storitev. Vse obstoječe spletne storitve je mogoče uporabljati brez sprememb in prilagoditev na novo proceduro, vendar novih ni mogoče objavljati s postopkom, opisanim v začetku tega poglavja. Še vedno je osnova za spletne storitve funkcijski modul, vendar je njegova objava zelo spremenjena. Dodana je funkcionalnost, da lahko sedaj več funkcijskih modulov združimo v enotno spletno storitev. Spletna storitev je po novem v sistemu SAP definirana kot objekt tipa Service Definition, ki je prikazan na sliki 4.11.



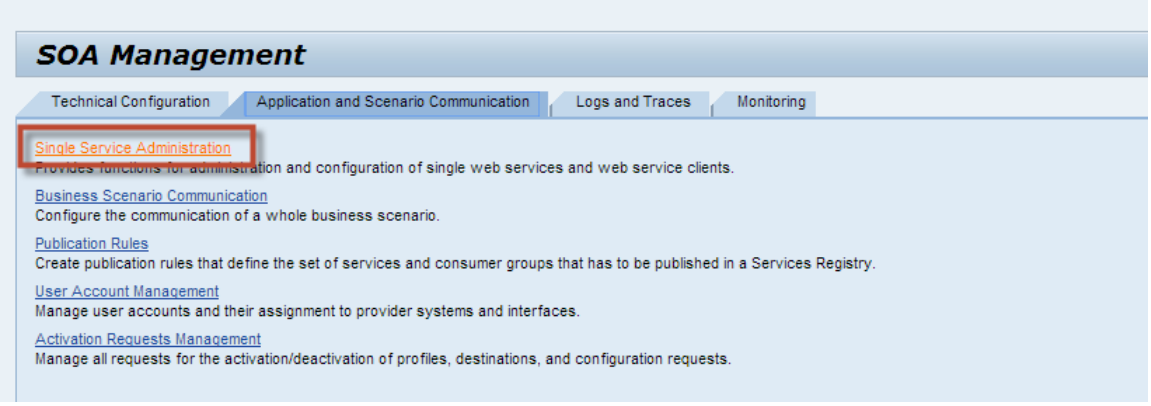
Slika 4.11: Primer kreacije objekta Service Definition

Naj opozorimo, da s kreacijo tega objekta to še ni klasična spletna storitev, do katere bi odjemalci že lahko dostopali. Treba je kreirati še konfiguracijo za objekt Service Definition, ki ga sestavljata dva objekta:

- Web service – klasična spletna storitev, do katere imajo dostop vsi odjemalci;
- Endpoint – vstopna točka za spletno storitev.

Kreacija konfiguracije se izvede s pomočjo transakcije SOAMANAGER v sistemu SAP. Transakcija se zažene v novem brskalniku in ne več v klasičnem SAP GUI. Da pridemo do nastavitev za naš objekt, moramo v meniju najprej izbrati

Application and Scenario Communication in podmeni Single Service Administration (slika 4.12), nato pa moramo preko iskalnika najti dotičen Service definition objekt.

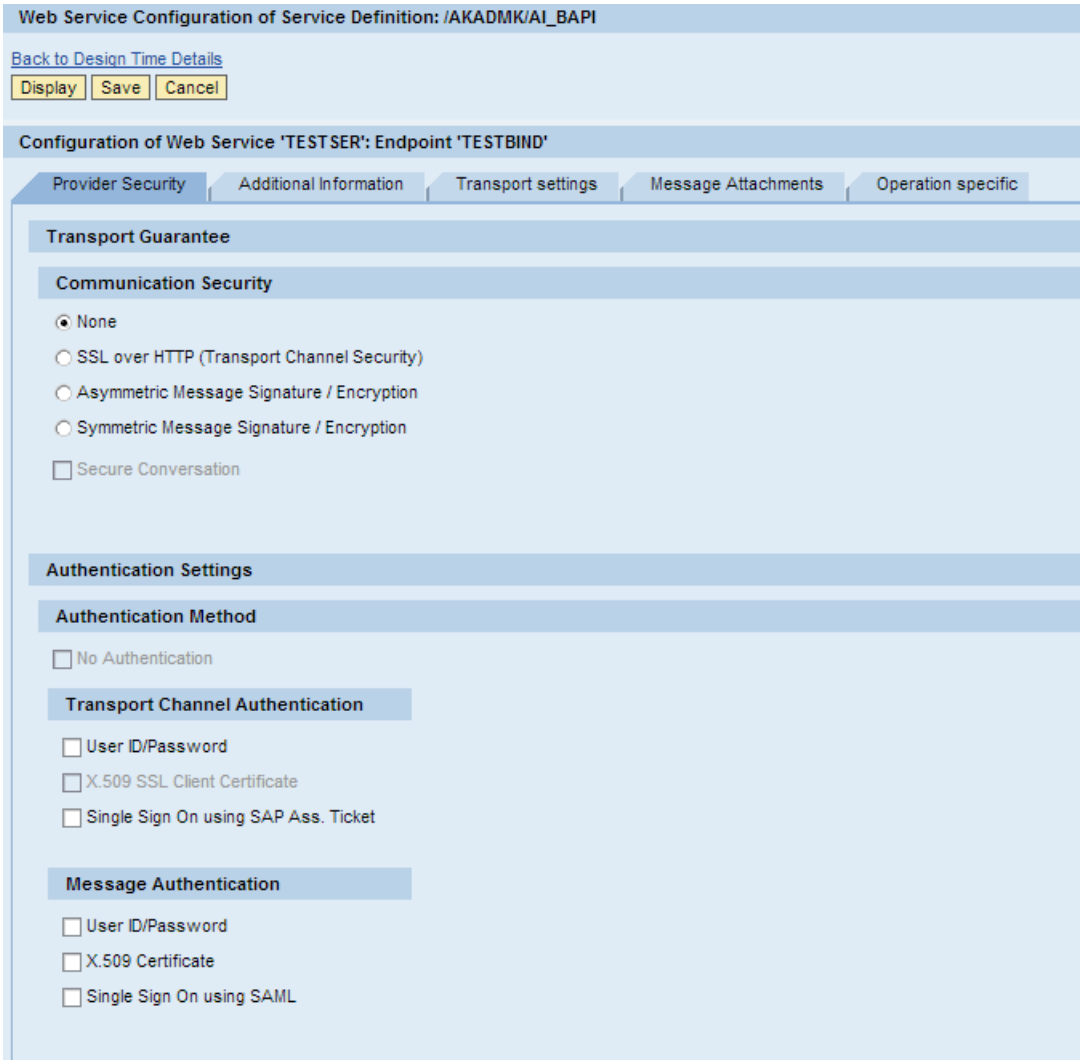


Slika 4.12: Transakcija SOAMANAGER, ki se prikazuje preko brskalnika

Ko najdemo objekt, lahko kreiramo oba elementa hkrati. Večina nastavitev v teh elementih že vsebuje privzete vrednosti samega sistema in se zelo malokrat spreminjajo, medtem ko se nekatere druge bolj prilagajajo sami uporabi spletne storitve. Med prilagodljivimi omenimo samo dve (slika 4.13):

- **nastavitev avtentikacije** (*angl. Authentication settings*)
  Avtentikacijo se lahko preverja na dveh nivojih: na nivoju transporta in na nivoju sporočila. Najpogostejše oblike avtentikacije so certifikat X.509 in prijavni podatki za sistem SAP (uporabniško ime in geslo). Sam sistem omogoča tudi prijavo preko SSO (Single sign on). Seveda se lahko avtentikacijo tudi izklopi, vendar to pomeni, da se bodo vsi klici izvajali pod posebnim uporabnikom znotraj sistema SAP, ki je definiran ob postavitvi sistema.

- **vrsta varnosti komuniciranja** (*angl. Communication security*)
  SAP podpira simetrično in asimetrično enkripcijo sporočil ali pa komunikacijo preko kanala SSL (Secure socket layer), vendar samo, če poteka avtentikacija preko certifikatov X.509.



Slika 4.13: Prikaz delnih nastavitev spletnega servisa

Vsaka definirana spletna storitev preko vmesnika ima lahko več konfiguracij, kar se izkaže za priročno, saj lahko klice prilagodimo skupinam odjemalcev tega vmesnika.

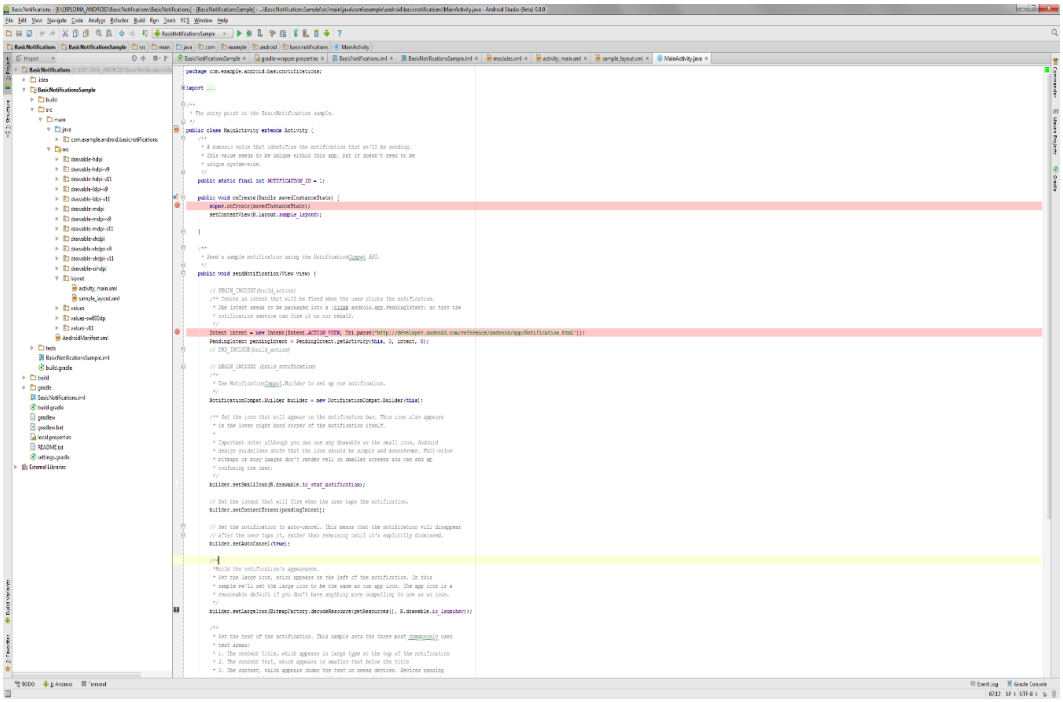
Poglavje 5

Mobilna aplikacija za opravila v SAP

5.1 Razvojno okolje Android studio

Za razvoj mobilne aplikacije smo uporabili razvojno okolje Android Studio (slika 5.1), ki temelji na IntelliJ IDEA podjetja JetBrains. Prvič je bilo predstavljeno v maju leta 2013 na Googlovi konferenci I/O. Trenutna verzija 0.8.0 je še vedno v beta različici. Android studio naj bi postal uradno okolje za razvoj na platformi Android in naj bi vseboval precej izboljšav v primerjavi z orodjem Eclipse ADT [26], ki so naslednje:

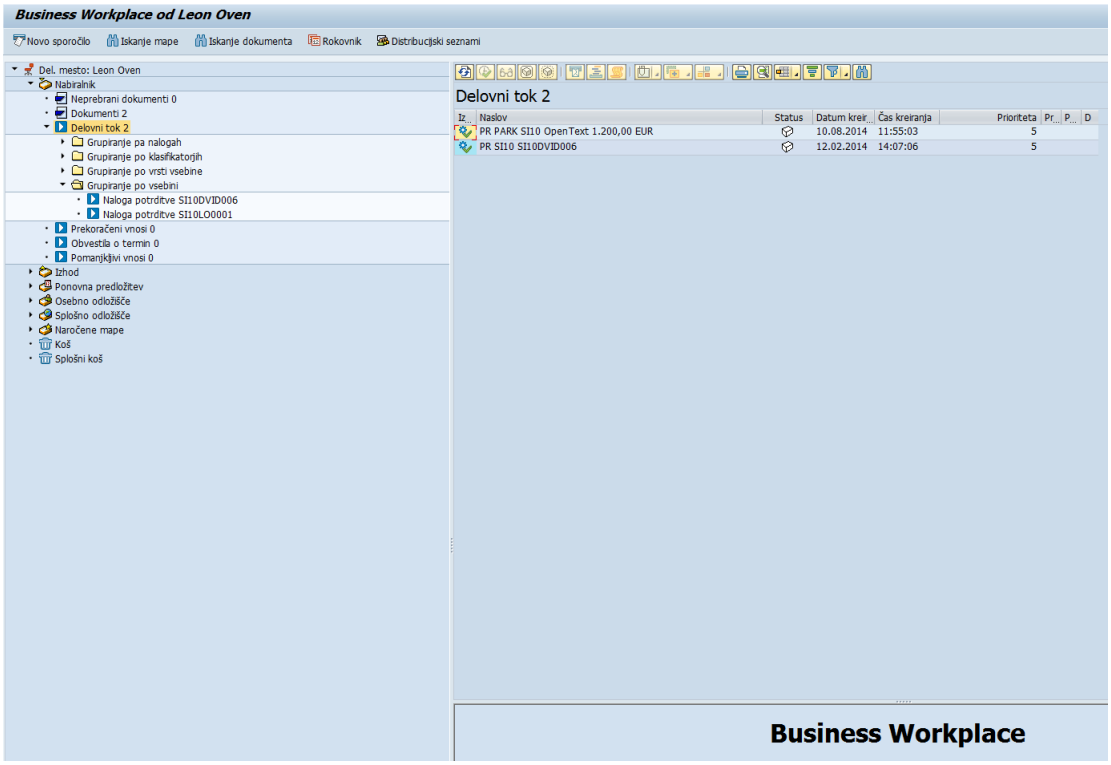
- vgrajena podpora za platformo v oblaku podjetja Google (*angl. Google Cloud Platform*);
- bogat grafični vmesnik s tematsko podporo;
- bogat nabor osnutkov kode za Google services in različne tipe naprav.



Slika 5.1: Razvojno okolje Android Studio

5.2 Koncept in opredelitev problema

Razviti želimo mobilno aplikacijo za prikaz opravil, ki jih uporabnik običajno vidi znotraj vhodnega nabiralnika (*angl. Business workplace*) v sistemu SAP. Vhodni nabiralnik je dostopen preko transakcije sbwp, njegovo vsebino pa vidimo na sliki 5.2.



Slika 5.2: Opravila vhodnega nabiralnika

Prikaz podatkov na mobilnih napravah bi pospešil odziv na opravila, predvsem pa bi dosegli takojšnjo obveščenost. Ker se kompleksnost opravil, ki se zbirajo znotraj vhodnega nabiralnika, razlikuje od procesa do procesa, moramo postaviti nekatere omejitve podpore, saj nekaterih funkcionalnosti v tem trenutku ni mogoče podpreti, druge pa so preobširne, da bi jih podprli v namene diplomske naloge.

Omejitve aplikacije so naslednje:

- ni integracije s preostalimi transakcijami v sistemu SAP;
- opravila predstavljajo preproste operacije.

V sklopu diplomske naloge smo podprli prikaz opravil procesa potrjevanje računov na mobilnih napravah. Ta ključen proces je pri podjetjih z vpeljanim sistemom SAP eden izmed prvih, ki ga želijo imeti podprtega v elektronski obliki. S tem namreč skrajšajo postopek potrjevanja in povečajo preglednost že potrjenih računov.

Opravila tega procesa tudi sodijo v sklop preprostih operacij, sploh če se osredotočimo samo na potrjevanje in zavračanje, ki sta najpogostejši opravili, ki jih opravljajo vodstveni kadri v podjetju.

Za potrebe delovanja mobilne aplikacije je potreben razvoj na obeh platformah: SAP in Android. Na platformi Android smo izdelali grafični vmesnik, ki se bo preko spletnih storitev, ki smo jih razvili za ta namen, povezoval s procesom znotraj sistema SAP. Za avtentikacijo smo uporabili kar standardni način z uporabniškim imenom in geslom sistema SAP.

5.3 Proces potrjevanja računov

Proces potrjevanja računov je zaporedje dogodkov nad posameznimi računi. Za potrebe diplomske naloge smo se odločili za primer potrjevanja, ki se pogosto pojavlja v podjetjih. Proces se začne s prihodom fizičnega dokumenta v podjetje in je lahko račun, avans ali dobropis. Dokument prejme enolično oznako (številka računa), preko katere se ga identificira v vseh fazah potrjevanja. V vložišču ta dokument skenirajo in pošljejo v računovodsko službo, kjer ga opremijo s posameznimi metapodatki, ki so za potrjevanje vsebinsko potrebni:

- tip dokumenta (račun, avans, dobropis ipd.);
- podjetje – podjetje, na katerega je naslovljen račun (pride v upoštev pri skupnih službah več podjetij);
- znesek računa;
- valuta računa;
- referenca – dobaviteljeva številka računa;
- datum storitve;
- datum prejema.

Poleg metapodatkov računa, računovodstvo izbere tudi scenarij, ki vsebuje zaporedne korake potrjevanja. Koraki potrjevanja predstavljajo posamezne vloge, ki morajo opraviti neko opravilo na dokumentu, preden se pot dokumenta zaključi. Za potrebe diplomske naloge smo uporabljali dva scenarija, ki se razlikujta v številu podpisnikov na posameznem dokumentu in sta razvidna iz tabele 5.1.

1 – Nivojsko potrjevanje	**2 – Nivojsko potrjevanje**
Vložišče	Vložišče
Računovodstvo	Računovodstvo
Podpisnik 1 nivo	Podpisnik 1 nivo
Knjigovodstvo	Podpisnik 2 nivo
	Knjigovodstvo

Tabela 5.1: Dva različna scenarija potrjevanja

Po izbiri scenarija mora računovodstvo račun zavesti v standardno aplikacijo SAP za račune in nato dokument poslati v potrjevanje (naslednji korak v izbranem scenariju). Vlogi Podpisnik 1 nivo in Podpisnik 2 nivo vsebujeta opravili, katerih izvedbo lahko prenesemo zunaj sistema SAP. Ti vlogi imata običajno možnost dokument potrditi ali pa ga dobavitelju zavrniti. Ti dve akciji smo v naši mobilni aplikaciji tudi podprli. Če vsi podpisniki račun potrdijo, gre ta v knjigovodstvo, kjer ga dokončno poknjižijo.

Celotno sosledje dogodkov in akcij, izvedenih na dokumentu v posameznih vlogah (zgodovina potrjevanja), se beleži na posebnem mestu in omogoča, da vsak uporabnik, ki opravilo prejme, lahko vidi, kdo je nad dokumentom že izvedel akcijo pred njim. Zgodovina potrjevanja, ki je enaka za vsa opravila znotraj delovnega nabiralnika, je uporabnikom pomembna in jo bomo zaradi tega razloga tudi prenesli v mobilno aplikacijo. Sestavljajo jo naslednji atributi:

- datum akcije;
- čas akcije;
- izbrani agent – agent, ki je bil izbran, da nastopi v določeni vlogi;

- dejanski agent – agent, ki je opravilo dejansko izvedel (če je izbrani agent odsoten, lahko opravilo izvede njegov zastopnik);
- izvedena akcija nad dokumentom (npr. parkiral, potrdil, zavrnil ipd.);
- vloga – vloga, v kateri je agent nastopal;
- komentar – vsak uporabnik ima možnost vpisa komentarja k dokumentu.

5.4 Razvoj spletnih storitev v SAP

Za potrebe mobilne aplikacije je treba iz sistema SAP zagotoviti funkcionalnosti za pridobivanje podatkov o opravilih. Glede na opis delovanja procesov in opis specifičnega procesa potrjevanja računov, lahko funkcionalnosti razdelimo na naslednje sklope:

- pridobivanje seznama opravil;
- pridobivanje podrobnih podatkov posameznega opravila;
- izvrševanje akcije za posamezno opravilo.

Pri razvoju posamezne funkcionalnosti smo se ravnali po postopkih, opisanih v podpoglavju 4.4.2. Pri kreiranju spletnega servisa smo sklopa Pridobivanje seznama opravil in Izvrševanje akcije za posamezno opravilo združili v isto spletno storitev kot dve ločeni metodi, saj se obe funkcionalnosti nanašata na splošne podporne funkcije za obvladovanje procesov v sistemu SAP. Vse spletne storitve so bile razvite na posebnem sistemu SAP, ki smo ga postavili za potrebe diplomske naloge.

5.4.1 Pridobivanje seznama opravil

Pridobivanje seznama opravil je enotna funkcionalnost za vse procese, podprte v sistemu SAP. Pri razvoju spletne storitve kot vhodni podatek potrebujemo uporabniško ime, na izhodu pa vrnemo seznam uporabnikovih opravil.

```
  IF own_tasks_only NE true.
*   Collect all agents that AGENT is a deputy of
    CALL METHOD /zcl_s_agents=>get_list_deputy_of
      EXPORTING
        is_agent          = agent
        i_date            = sy-datum
        i_active          = true
        i_passive         = space
      IMPORTING
        et_substitutions = lt_subs.

*   Get tasks as deputy
    LOOP AT lt_subs ASSIGNING <sub>.
      MOVE-CORRESPONDING <sub> TO agent.
      PERFORM append_tasks_of_agent
              USING agent project role
              CHANGING lt_tasks.

    ENDLOOP. " AT lt_subs ASSIGNING <sub>.
  ENDIF. " own_tasks_only NE true.
  SORT lt_tasks BY tskid.
  {...}
  LOOP AT lt_tasks ASSIGNING <task>.
    wa_task-task_id        = <task>-tskid.
    wa_task-project        = <task>-project.
    wa_task-unique_id      = <task>-unqid.
    wa_task-role           = <task>-role.
    wa_task-agent_type     = <task>-otype.
    wa_task-agent_id       = <task>-objid.
    wa_task-reference_id   = <task>-refid.
    wa_task-created_date   = <task>-erdat.
    wa_task-created_time   = <task>-erzet.
    wa_task-workitem_text = <task>-wi_text.
    APPEND wa_task TO task_list.
  ENDLOOP. " AT lt_tasks ASSIGNING <task>.
```

Seznam 5.1: Del funkcijskega modula za pridobivanje opravil uporabnika

5.4.2 Pridobivanje detajlnih podatkov posameznega opravila

Ko želi uporabnik videti podrobnosti opravila, je treba prebrati podatke dokumenta, na katerega se opravilo nanaša. Te vrste funkcij moramo razviti za vsak proces posebej, saj ima vsak proces svoje lastne specifične podatke. Podatke, ki smo jih za naše potrebe prenesli v mobilno aplikacijo, smo že opisali v podpoglavju 5.3.

Na vhodu funkcijskega modula bomo potrebovali naslednje parametre:

- številka računa – enolična številka dokumenta;
- jezik – jezik za pravilen prikaz opisov;
- šifra podjetja.

```
{...}
 ps_headerdata-bline_date          = wa_hd-zfbdt.
 ps_headerdata-due_date            = wa_hd-netdt.
 ps_headerdata-tax_date            = wa_hd-vatdate.
WRITE wa_hd-belnr        TO  ps_headerdata-doc_no.
WRITE wa_hd-rbelnr       TO  ps_headerdata-inv_doc_no.
WRITE wa_hd-lifnr        TO  ps_headerdata-vendor.

CALL FUNCTION 'REF_DOC_NO_CONVERSION_OUTBOUND'
  EXPORTING
    i_ref_doc_no_long = wa_hd-xblnr
  IMPORTING
    e_ref_doc_no      = ps_headerdata-ref_doc_no
    e_ref_doc_no_long = ps_headerdata-ref_doc_no_long.
 {...}
CALL FUNCTION 'BAPI_CURRENCY_CONV_TO_EXTERNAL'
  EXPORTING
    currency        = wa_hd-waers
    amount_internal = wa_hd-wrbtr
  IMPORTING
    amount_external = ps_headerdata-gross_amnt.
```

Seznam 5.2: Del kode za branje podatkov o posameznem opravilu

5.4.3 Izvrševanje akcije za posamezno opravilo

Nad vsakim opravilom, ki ga prikažemo, lahko uporabnik izvede določeno akcijo. Za izvedbo opravila smo implementirali funkcijski modul, ki poskrbi za izvedbo obstoječe logike v standardnem procesu sistema SAP in ima naslednje vhodne podatke:

- TASK_ID – enolična številka opravila;
- BUTTON – izbrana akcija uporabnika;
- AGENT – oznaka uporabnika v sistemu SAP, ki opravilo izvaja;
- COMMENT – komentar uporabnika ob izvedbi akcije.

Funkcijski modul kot rezultat vrne napako, če klic ni bil uspešen, sicer pa ne vrne nobene vrednosti.

```
{...}
 IF gv_multi_history_insert NE true.
   CALL METHOD read_history_into_buffer.
 ENDIF.
 wa_ap-entid      = LINES( gs_history_buffer-history ) + 1.
 wa_ap-role       = i_role.
 wa_ap-act_date   = i_date.
 wa_ap-act_time   = i_time.
 wa_ap-action     = i_action.
 ls_agent         = get_forwarded_to_agent( ).
 wa_ap-fwd_otype  = ls_agent-otype.
 wa_ap-fwd_objid  = ls_agent-objid.
 INSERT wa_ap INTO TABLE gs_history_buffer-history.
   CALL METHOD zcl_s_framework=>check_comment
     EXPORTING
       it_comment       = it_comment
       i_min_total_len = 1
     EXCEPTIONS
       too_short        = 1
       OTHERS           = 2.
```

Seznam 5.3: Del kode funkcijskega modula za izvedbo akcije

5.5 Razvoj aplikacije

Z mobilno aplikacijo za prikazovanje opravil smo želeli uporabnikom omogočiti njihovo izvajanje tudi tedaj, ko niso prisotni na svojem delovnem mestu oziroma nimajo dostopa do standardnega programa SAP GUI, ki ga lahko poganjamo samo na osebnih računalnikih. Uporabnik z vstopom v aplikacijo pride do seznama opravil, s klikom na posamezno opravilo pa se mu prikaže okno s podrobnostmi z dvema zavihkoma, podatki in zgodovina. Na zavihku podatki se vidijo vsi metapodatki dokumenta (seznam se nahaja v podpoglavju 5.3), dodani pa sta še akciji potrdi in zavrni. Na zavihku zgodovina so vidni podatki revizijske sledi dokumenta, skupaj s komentarji. Aplikacija vse podatke pridobiva iz sistema SAP, s katerim komunicira preko protokola SOAP. Na sliki 5.3 je prikazana arhitektura sistema. Pri zasnovi aplikacije smo upoštevali, da bo aplikacijo na uporabniške telefone prenašal administrator, ki bo tudi poskrbel za nastavitev identifikacije uporabnika, zato vpisa uporabnikov ni treba implementirati.

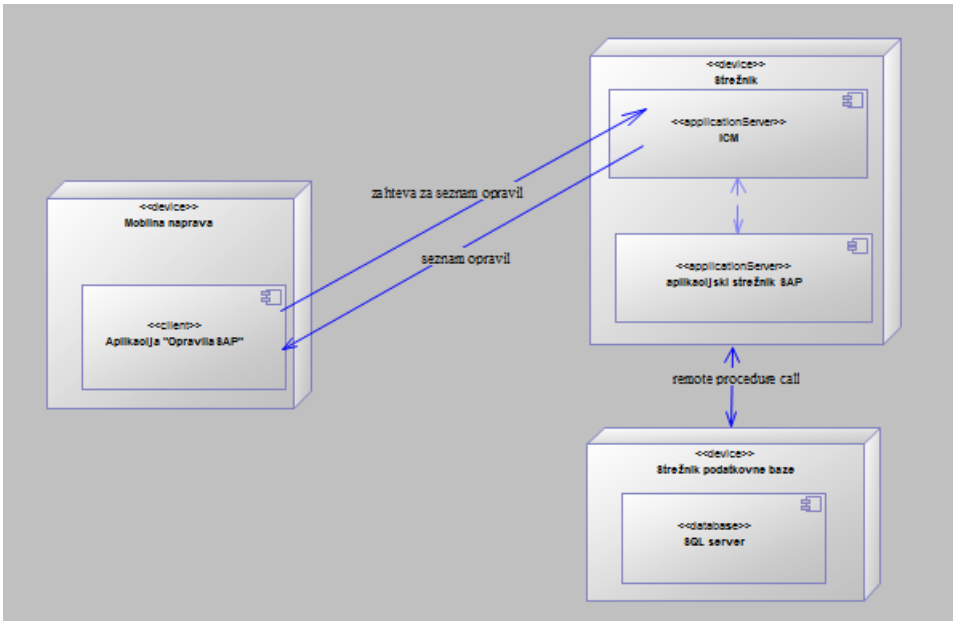
Celoten razvoj mobilne aplikacije je potekal v okolju Android Studio. Razvoj celotne aplikacije lahko razdelimo v dva sklopa:

- komunikacija s sistemom SAP;
- grafični vmesnik.

Vsak sklop posebej bomo podrobno opisali v nadaljevanju.

5.5.1 UML diagram postavitve

Za implementacijo aplikacije smo potrebovali poleg mobilne naprave tudi sistem SAP s pripadajočo podatkovno bazo. Vse zahteve, ki jih mobilna naprava pošlje sistemu SAP, prestreže enota ICM, ki te zahteve nato posreduje aplikacijskemu strežniku v obdelavo. Postavitev takšnega sistema je prikazana na sliki 5.3.



Slika 5.3: UML diagram postavitve

5.5.2 Komunikacija s sistemom SAP

Za klicanje spletnih storitev, ki smo jih razvili in predstavili v prejšnjem podpoglavju, smo kot pomoč uporabili odprtokodno knjižnico ksoap2-android verzije 2.4 [27]. Knjižnica olajša ustvarjanje sporočil SOAP in klicanje spletnih storitev. Celotno komunikacijo s spletnimi storitvami smo implementirali v javanskem razredu SoapUtil, v katerem smo ustvarili dve glavni metodi:

- CreateRequest in
- CallSoap.

Metoda CallSoap skrbi za klicanje spletnih storitev, medtem ko z metodo CreateRequest (seznam 5.4) ustvarimo sporočilo glede na metodo in spletno storitev, ki jo želimo klicati.

```
public static SoapObject CallSoap(SoapObject request, String
    SOAP_ACTION, String URL) {
        //Declare the version of the SOAP request
        SoapSerializationEnvelope envelope = new
            SoapSerializationEnvelope(SoapEnvelope.VER11);
        envelope.setOutputSoapObject(request);
        envelope.dotNet = false;

        HttpTransportSE androidHttpTransport = new
            HttpTransportSE(URL);
        SoapObject resultSOAP = null;
        try {
            List<HeaderProperty> headers = GetHeader();
            androidHttpTransport.debug = true;
            //this is the actual part that will call the
                webservice
            androidHttpTransport.call(SOAP_ACTION, envelope,
                headers);

            // Get the SoapResult from the envelope body.
            resultSOAP = (SoapObject) envelope.bodyIn;

        } catch (Exception e) {
            e.printStackTrace();
        }
        return resultSOAP;
    }
```

Seznam 5.4: Metoda za klicanje spletnih storitev CallSoap

Za shranjevanje opravil smo kreirali javanski razred OpraviloSAP, ki vsebuje vse podatke o posameznem opravilu. Ker smo želeli celotno zadevo poenostaviti in narediti uporabno tudi z uporabo dinamičnih klicev, smo ustvarili metodo

setProperty (seznam 5.5). Ta kot parametra sprejme:

- PropertyName – označitev imena podatka, ki ga želimo spremeniti;
- Value – vrednost, ki jo želimo temu podatku nastaviti.

Z omenjeno metodo smo poenostavili prenašanje vrednosti posameznih podatkov iz povratnega sporočila spletne storitve v posamezen objekt tipa OpraviloSAP, kot je razvidno iz seznama 5.6. Pogoj delovanja omenjene metode je enakoimensko poimenovanje razrednih atributov z atributi spletnega sporočila.

```
public void setProperty(String PropertyName, String Value) {
        Class clas = this.getClass();
        Field f1;
        if(Value.equals("anyType{}"))
            return ;
        try {
            f1 = clas.getField(PropertyName);
            f1.set(this, Value);
        } catch (NoSuchFieldException e) {
            //e.printStackTrace();
        } catch (IllegalAccessException e) {
            //e.printStackTrace();
        }
    }
```

Seznam 5.5: Metoda setProperty razreda OpravilaSAP

```
SoapObject resultSOAP = SOAPUtil.CallSoap(request, SOAP_ACTION,
    Configuration.URL);
// parse results
SoapObject tasks = (SoapObject) resultSOAP.getProperty(3);
SOAP_ACTION = Configuration.NAMESPACE + Configuration.
    METHOD_GET_DETAIL;
for (int i = 0; i < tasks.getPropertyCount(); i++) {
      SoapObject task = (SoapObject) tasks.getProperty(i);
       OpraviloSAP item = new OpraviloSAP();

       for (int j = 0; j < task.getPropertyCount(); j++) {
                        PropertyInfo pi = new PropertyInfo();//=
                            (PropertyInfo) task.getProperty(j);
                        task.getPropertyInfo(j, pi);
                        //SoapPrimitive pi = (SoapPrimitive)
                            task.getProperty(j);
                        item.setProperty(pi.getName(), pi.
                            getValue().toString());
       }

       ITEMS.add(item);
       ITEM_MAP.put(item.TASK_ID, item);
}
```

Seznam 5.6: Primer dinamičnega nastavljanja vrednosti objekta

Ker v aplikacijah android ni mogoče klicati spletnih storitev v glavnih aktivnostih, je treba kreirati nov razred, ki razširja razred AsyncTask, s katerim omogočimo klicanje spletnih storitev v ozadju [28]. Glavni metodi tega razreda, ki ju mora razvijalec implementirati, sta doInBackground, v kateri izvajamo aktivnosti v ozadju, in metoda onPostExecute, ki se kliče po končani metodi doInBackgroud in lahko posega v niti grafičnega vmesnika. Tako lahko v metodi onPostExecute podatke, pridobljene s klicem spletne storitve, prikažemo na ekranu.

Za shranjevanje podatkov zgodovine smo kreirali javanski razred Zgodovina, v katerem smo tudi implementirali metodo setProperty. Posamezni objekt, Zgodo-

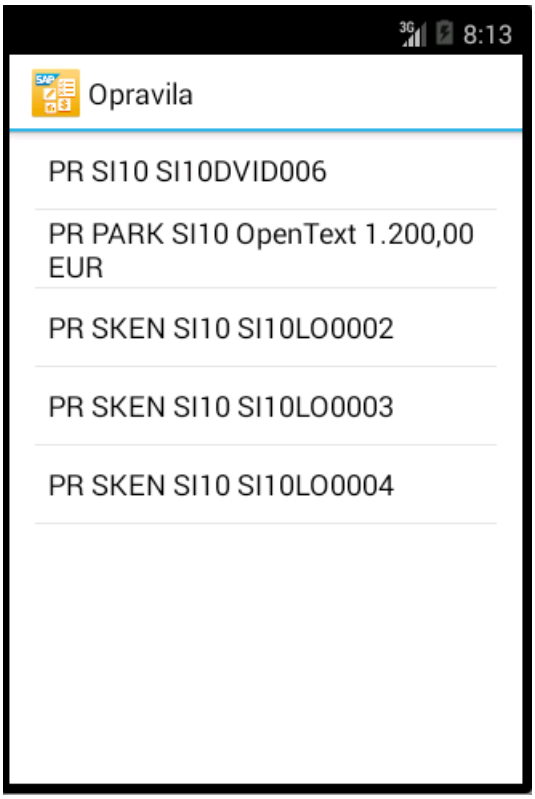
vina, vsebuje podatke o enkratnem posegu v dokument, tako da celotno zgodovino predstavlja zbirka teh objektov tipa Array. Za prikaz tega seznama na ekranu smo morali implementirati svoj razred ZgodovinaAdapter (seznam 5.7), ki razširja osnovni razred BaseAdapter in služi za prikazovanje objektov na seznamu. S to razširitvijo je mogoče kot element seznama uporabiti poljuben videz, ki ga kreiramo v sklopu projekta.

```
public View getView(int position, View convertView, ViewGroup
    parent) {
        ViewHolder holder;
        if (convertView == null) {
            convertView = layoutInflater.inflate(R.layout.
                listview_item, null);
            holder = new ViewHolder();
            holder.roledescrView = (TextView) convertView.
                findViewById(R.id.txtVlogaZGO);
            holder.agentView = (TextView) convertView.
                findViewById(R.id.txtAgentZGO);
            holder.akcijaView = (TextView) convertView.
                findViewById(R.id.txtAkcijaZGO);
            holder.datetimeView = (TextView) convertView.
                findViewById(R.id.txtDatumUraZGO);
            holder.ReasonView = (TextView) convertView.
                findViewById(R.id.txtReasonZGO);
            convertView.setTag(holder);
        } else {  holder = (ViewHolder) convertView.getTag(); }
        Zgodovina Item = (Zgodovina) listData.get(position);
        holder.roledescrView.setText(Item.getROLEDESCR());
        holder.akcijaView.setText(Item.getACTION_TEXT());
        holder.datetimeView.setText(Item.getACTUAL_DATE() + "␣"
            + Item.getACTUAL_TIME());
        holder.agentView.setText(Item.getACTUAL_USER());
        holder.ReasonView.setText(Item.getREASON());
        return convertView;
    }
```

Seznam 5.7: Glavna metoda razreda ZgodovinaAdapter

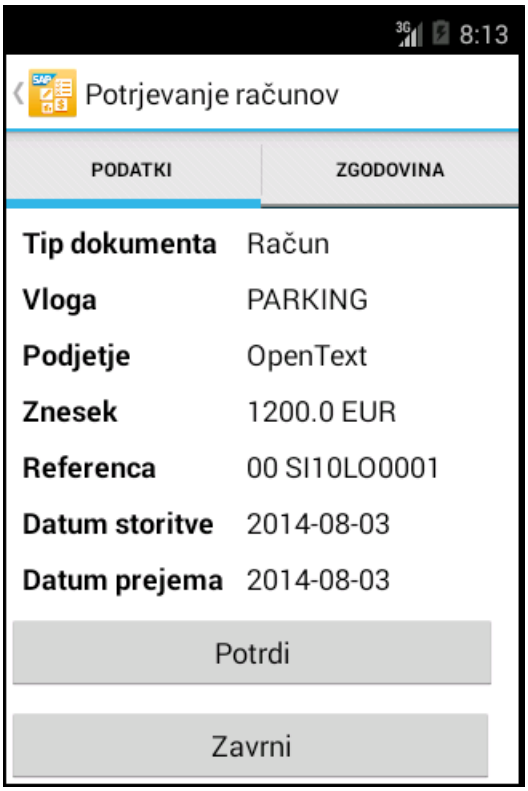
5.5.3 Grafični vmesnik

Pri grafičnem vmesniku smo zaradi podprtosti samo enega procesa v aplikaciji izpustili ekran, na katerem bi uporabnik izbral proces, katerega opravila želi videti. Prav tako smo se za obstoječo rešitev odločili, ker lahko uporabnik na ta način vidi vsa svoja opravila, ne glede na proces, na katerega se nanašajo. Tako se uporabniku po zagonu aplikacije prikaže seznam vseh njegovih opravil procesa potrjevanja računov (slika 5.5). Seznam opravil je implementiran s pomočjo razreda ListFragment [29], ki vsebuje preddefiniran videz, stil in povezovanje podatkov. Vsaka vrstica seznama predstavlja eno opravilo. Pri vprašanju, katere podatke posameznega opravila uporabniku prikazati na tem seznamu, smo se odločili, da prikažemo glavni metapodatek imenovan workitemtext, ki je viden tudi uporabnikom v vhodnem nabiralniku sistema SAP.



Slika 5.4: Seznam opravil

Po kliku na posamezno opravilo se prikaže ekran s podrobnostmi, ki smo ga razdelili na dva zavihka, podatki in zgodovina (slika 5.5). Zavihek podatki prikazuje glavne podatke in akciji, ki ju uporabnik lahko izvede nad dokumentom.



Slika 5.5: Zavihek podatki z glavnimi metapodatki računa

Za vsak gumb je treba kreirati dogodek OnClickListener, ki se izvede ob kliku na gumb (kot denimo na seznamu 5.8). V našem primeru smo klicali metode spletne storitve za izvajanje akcije. V spletno storitev smo tako na podlagi izbire gumba poslali ustrezno akcijo skupaj s podatki o uporabniku. Po uspešnem klicu uporabniku prikažemo sporočilo, da je bila akcija uspešno izvedena. To sporočilo je tipa Toast, ki je po definiciji kratko povratno sporočilo, ki se prikaže v majhnem modalnem oknu na dnu zaslona (slika 5.6).

```
bPotrdi.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
    if(CallActionSap(bPotrdi.getHint().toString())) {
                Toast msg = Toast.makeText(rootView.getContext()
                    ,
                "Akcija je bila uspesno izvedena", Toast.LENGTH\
                    LONG);
        msg.show();
        activity2 = (OpraviloDetailActivity) getActivity();
        activity2.navigateUpTo(new Intent(activity2,
            OpraviloListActivity.class));
        }
    }
});
```

Seznam 5.8: Implementacija gumba Potrdi

Slika 5.6: Prikaz sporočila Toast po uspešno izvedeni akciji

Na zavihku zgodovina prikazujemo podatke revizijske sledi (slika 5.7). Razvoj tega dela je bil glede videza najbolj zahteven, saj je bilo potrebnega kar nekaj oblikovalskega znanja. V prid razvijalcem ne gre niti to, da ni veliko oblikovalskih ogrodij, s katerimi bi oblikovali celotno aplikacijo, kot jih poznamo za spletne strani.

Slika 5.7: Prikaz sporočila Toast po uspešno izvedeni akciji

Poglavje 6

Sklepne ugotovitve

Danes na trgu obstaja kar nekaj rešitev za integracijo sistema SAP z mobilnimi napravami. Večino teh rešitev ponuja mobilno platformsko neodvisne rešitve, ki omogočajo podjetjem tudi določeno prilagodljivost. Med vsemi temi rešitvami je največkrat uporabljena prav rešitev podjetja SAP, katerega cilj do konca desetletja je poleg storitev v oblaku tudi razvoj rešitev za mobilne naprave. Seveda pa te rešitve niso poceni in tako se odpirajo možnosti tudi preostalim ponudnikom, ki niso specializirali svojih rešitev. Ko pogledamo preostale možnosti za razvoj mobilnih aplikacij, ugotovimo, da je mogoče z nekaj prilagoditvami na procesih v SAP hitro pripeljati del ali pa kar celotni proces tudi na mobilne naprave. Tako nam je za platformo Android z uporabo protokola SOAP uspelo razviti mobilno aplikacijo za prikazovanje opravil procesa potrjevanja računov. Pri razvoju aplikacije smo dobro spoznali platformo Android, tako da bi v primeru implementacije nove rešitve ta potekala še hitreje. Prav tako smo spoznali, kako preprosto in s funkcijami bogato je novo okolje za razvoj tovrstnih aplikacij podjetja Google.

Z uporabo nekega vmesnega strežnika, na katerem bi implementirali storitve s protokolom REST in bi tako v aplikaciji uporabili te storitve in ne storitev SAP preko protokola SOAP, bi lahko dodatno zmanjšali obseg prometa iz okolja podjetja do mobilne naprave. V takem primeru bi storitve na dodatnem strežniku do storitev sistema SAP dostopale preko protokola SOAP. V aplikaciji bi se kot izboljšavo lahko podprlo več procesov, prav tako pa bi lahko razširili tudi funkcionalnosti na obstoječi aplikaciji, s čimer bi prav gotovo zajeli še večji nabor uporabnikov, ki bi

jim omenjena rešitev lahko pomagala za hitrejšo odzivnost in večjo produktivnost.

Literatura

[1] (maj 2014) SAP history, dostopno na: http://global.sap.com/corporate-en/our-company/history/index.epx.

[2] Jose Antonio Hernandez, *SAP R/3 handbook*. The McGraw−Hill Companies, Inc. 1997.

[3] (maj 2014) Donald K. Burleson, *Oracle SAP Administration*, O'Reilly Media 1st Edition 1999, dostopno na: http://oreilly.com/catalog/sapadm/chapter/ch01.html.

[4] (maj 2014) Mobile: Native Apps, Web Apps, and Hybrid Apps, dostopno na: http://www.nngroup.com/articles/mobile-native-apps.

[5] Eyal Katz, Jan Rumig, *How to Write Online Mobile Apps Consuming SAP Back End Systems*, SAP documentation, 2012.

[6] (maj 2014) Native Mobile App, dostopno na: http://www.techopedia.com/definition/27568/native-mobile-app.

[7] S. Olson, J. Hunder, B. Horgen, K. Goers, *Professional Cross-Platform Mobile Development in* , Indiana: John Wiley & sons, Inc., 2012.

[8] (maj 2014) When to Go Native, Mobile Web or Cross-Platform/Hybrid, dostopno na: http://tech.pro/blog/1355/when-to-go-native-mobile-web-or-cross-platformhybrid.

[9] (maj 2014) Mobile Marketing Statistics 2014, dostopno na: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics.

[10] (maj 2014) Neptune sofware, dostopno na: http://neptune-software.com.

[11] James Wood, *ABAP Cookbook* , SAP Press 2010, pogl. 1.

[12] Myer T., *Begining PhoneGap*, Indiana, John Wiley & sons, Inc., 2012, pogl.1,2,3.

[13] (junij 2014) Native JSON, dostopno na: http://en.wikipedia.org/wiki/JSON.

[14] (junij 2014) Movillizer, dostopno na: http://movilizer.com.

[15] (junij 2014) The Gartner MADP Conundrum, dostopno na: http://www.mobilesmith.com/the-gartner-madp-conundrum.

[16] (junij 2014) Sap Road Map for Mobile Apps, dostopno na: https://websmp101.sap-ag.de/ sapidb/011000358700000995732011E.com.

[17] Carsten Bönnen, Volker Drees, André Fischer, Ludwig Heinz, Karsten Strothmann, *OData and SAP NetWeaver Gateway*, SAP Press 2014, pogl. 1, 2.

[18] (junij 2014) SOAP, dostopno na: http://en.wikipedia.org/wiki/SOAP

[19] (avgust 2014), SAPUI5, dostopno na:http://scn.sap.com/community/developer-center/front-end/blog/2013/12/11/what-is-openui5-sapui5.

[20] (avgust 2014), Get to Know the UI Development Toolkit for HTML5, dostopno na: http://scn.sap.com/docs/DOC-31625&jive_content_id_Overview.

[21] (junij 2014) SOAP Introduction, dostopno na: http://www.w3schools.com/webservices/

[22] (junij 2014) Zakaj spletne storitve niso zgolj novi komponentni model, dostopno na: http://www.soa.si/juric/zakajSpletne.pdf

[23] (junij 2014) Storitvene arhitekture, ki temeljijo na modelih, dostopno na: http://cot.uni-mb.si/cotl/zima2003/juric.html.

[24] (junij 2014)) SOAP Version 1.2 Part 1:Messaging Framework (Second Edition), dostopno na: http://www.w3.org/TR/soap12-part1.

[25] Tanmaya Gupta, *Function Modules in ABAP, A Quick Reference Guide*, SAP Press 2014, poglavje 1,5,6.

[26] (junij 2014) Android studio, dostopno na:
https://developer.android.com/sdk/installing/studio.html.

[27] (junij 2014) KSoap2-android library, dostopno na:
https://code.google.com/p/ksoap2-android.

[28] (junij 2014) AsyncTask class, dostopno na:
http://developer.android.com/reference/android/os/AsyncTask.html.

[29] (junij 2014) Fragment class, dostopno na:
http://developer.android.com/reference/android/app/Fragment.html.